PAC-C3D
A New Software Architectural Model
for Designing 3D Collaborative Virtual Environments

Thierry Duval, Cédric Fleury

IRISA – Rennes – France

➢ Designing 3D CVE is a complex task :

   ✔ Address 3D interaction and immersion issues
      ✗ Adaptation to 3D graphics API, to various displays (desktop, tablets, CAVE, …)

   ✔ Address collaborative issues:
      ✗ Distribution, synchronization, consistency maintenance of the shared data

➢ To meet all these requirements, merge :

   ✔ HCI and CSCW results about independence to graphics 2D API

   ✔ 3D CVE results about collaborative issues

- Related work
  - ✔ Distributed architectures for 3D CVE
  - ✔ HCI and CSCW software architectural models

- The PAC-C3D model
  - ✔ Independence to 3D graphics
  - ✔ Adaptation to distribution modes

- PAC-C3D main features
  - ✔ Drive the same abstraction with different input devices
  - ✔ Visualize the same abstraction with different 3D API
  - ✔ Delegate behavior to abstraction and other presentations
  - ✔ Make interoperability possible between heterogeneous 3D viewers
  - ✔ Maintain consistency between Virtual and Physical world

- Conclusion and future work

➢ 3D CVE architectures : duplicated
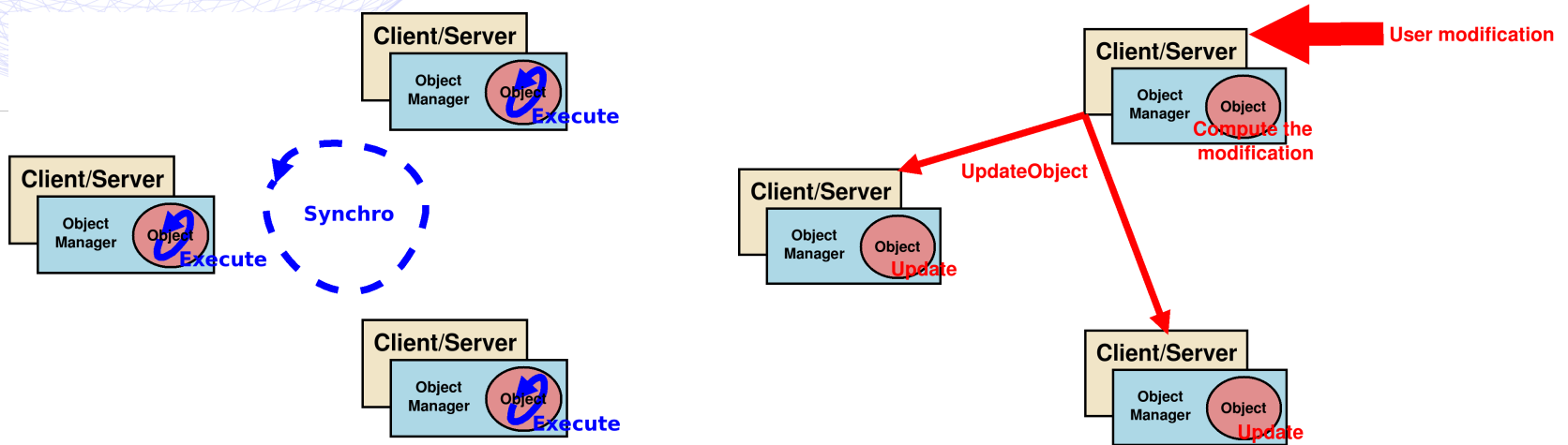
➤ 3D CVE architectures : centralized
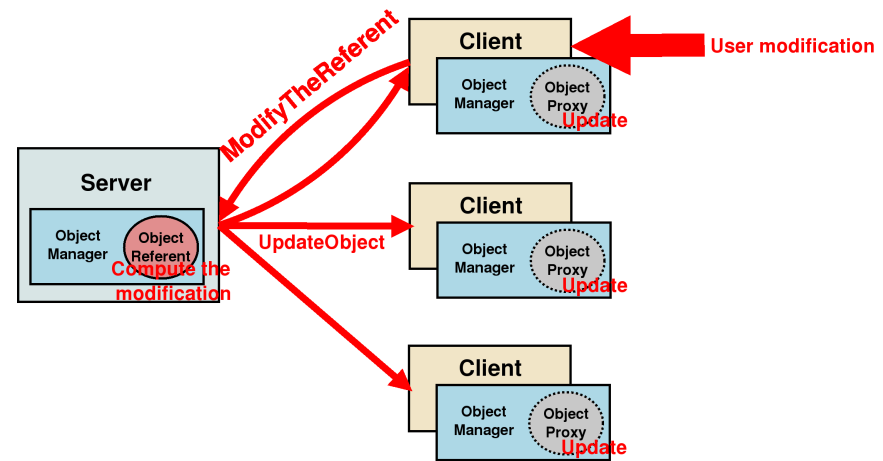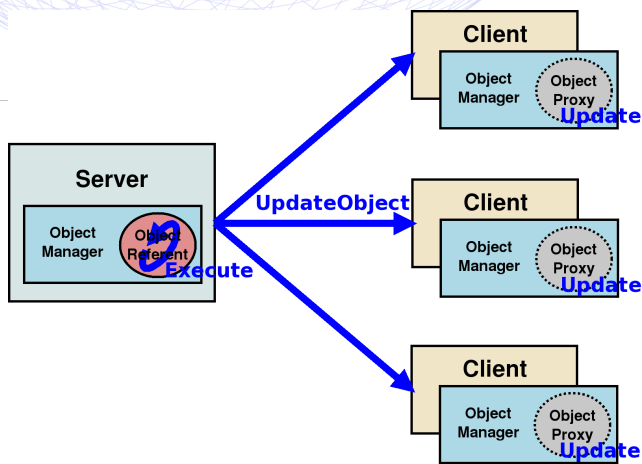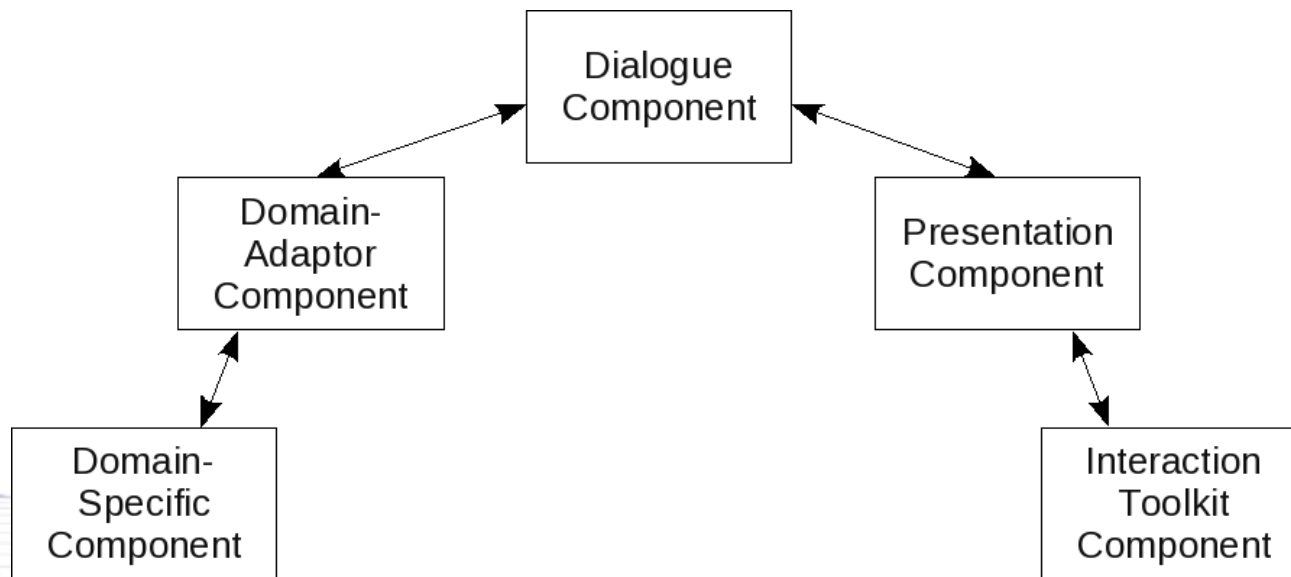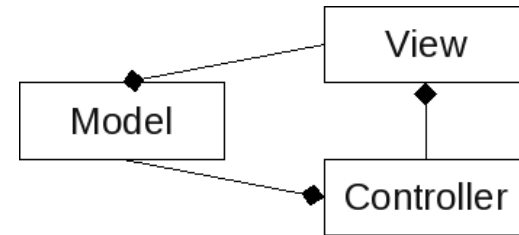
- 3D CVE architectures : hybrid
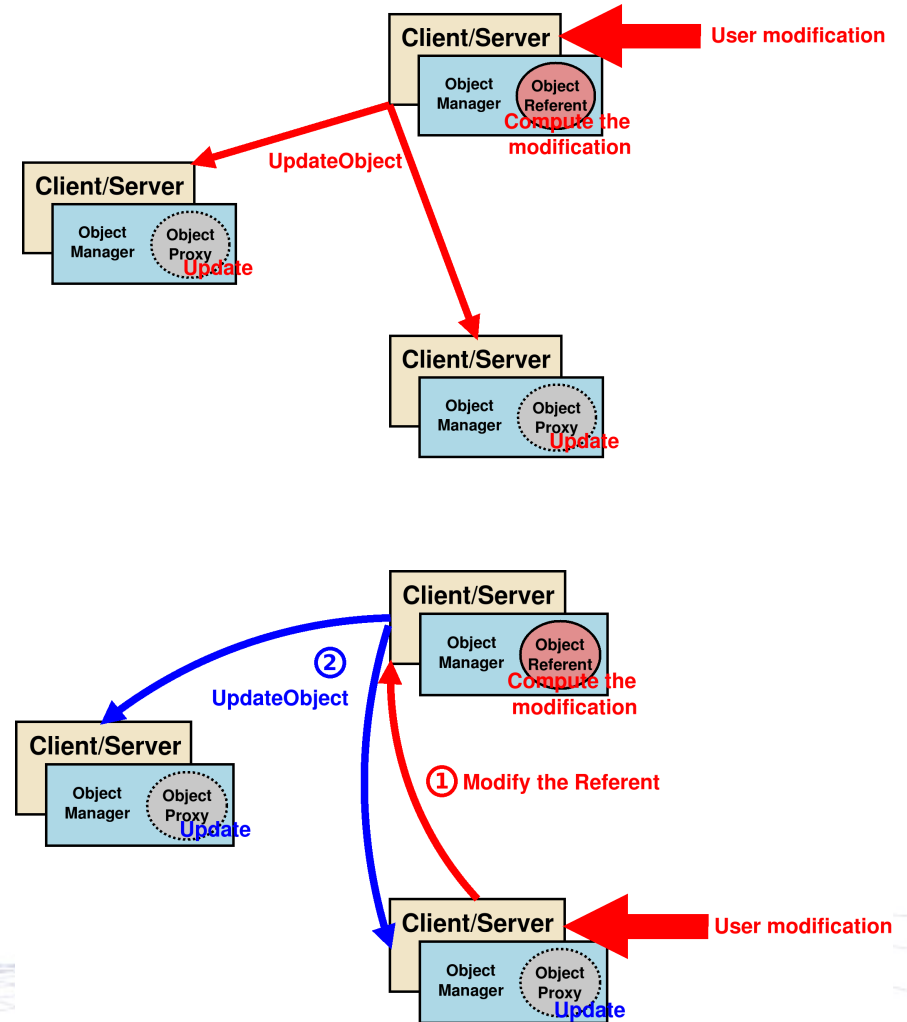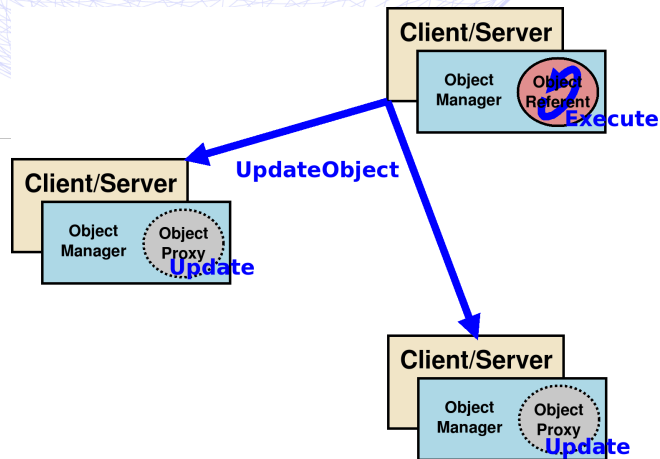
➢ HCI models

 ✔ PAC [Coutaz 1987] / MVC [Reenskaug 1979] [Goldberg 1990] / Arch [UIMS 1992]
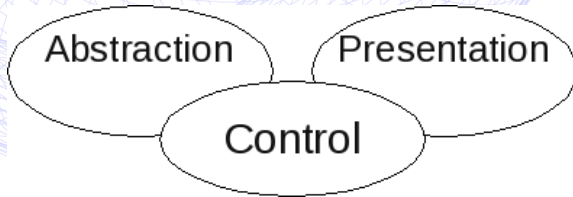
- CSCW models
  - [Ellis, Wainer 1994] ontological, coordination and user-interface models
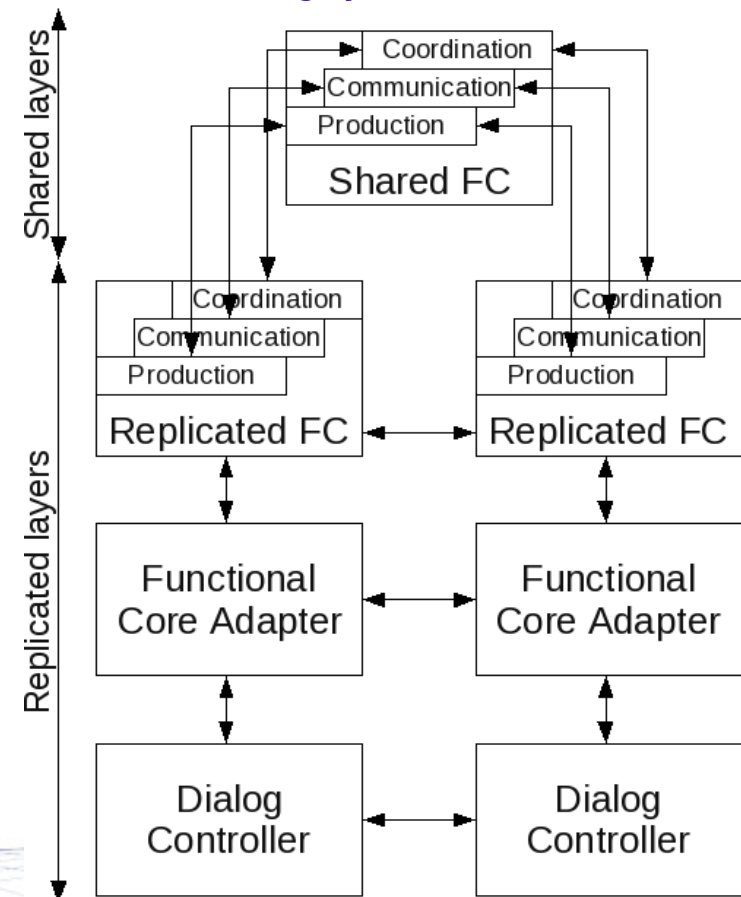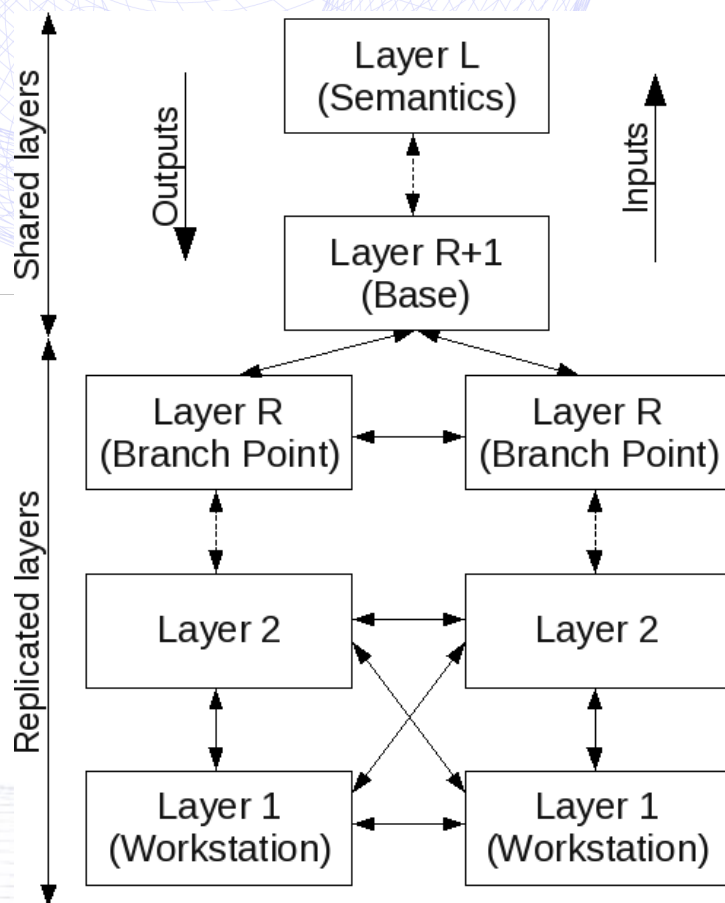 - Clover concept, PAC* [Calvary, Coutaz, Nigay 1997]
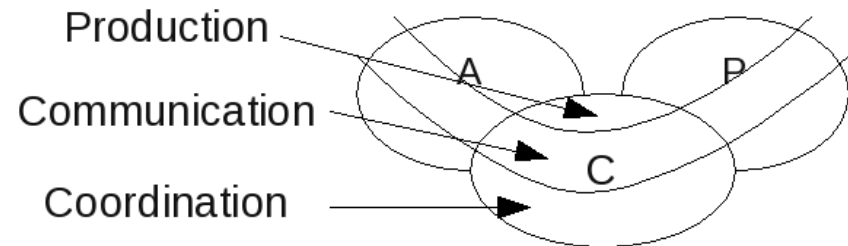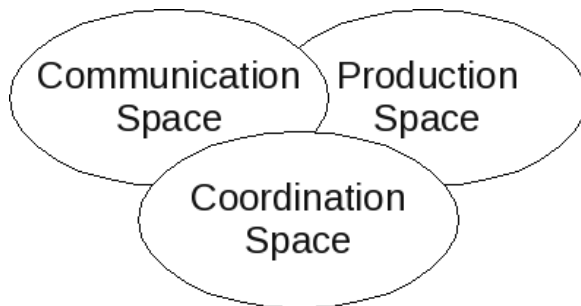
- CSCW models
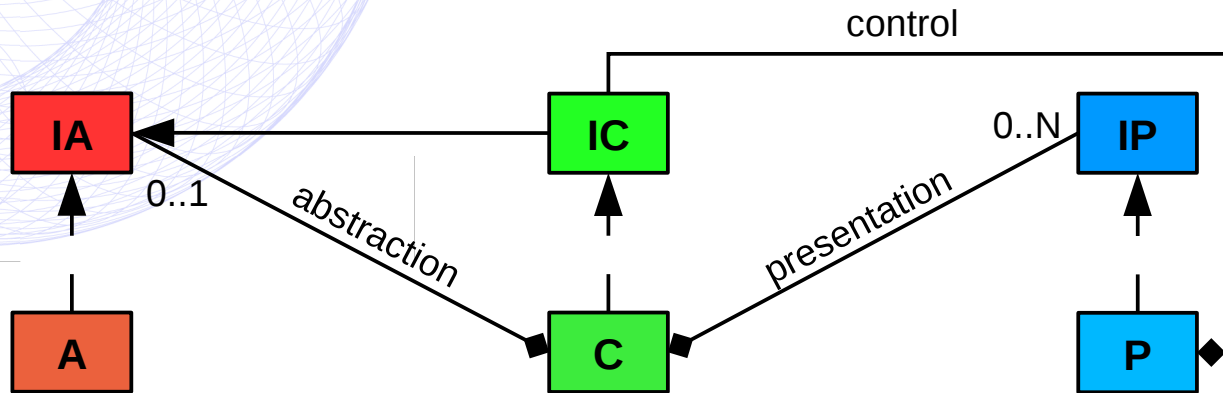 - ✔ Dewan's model [Dewan 1999], Clover model [Laurillau, Nigay 2002]

➢ The need : help for designing 3D CVE

   ✔ That should not rely too much on the 3D graphics API

   ✔ That should not be too dependent from the real displays

   ✔ That should not be too dependent from the network issues

➢ First solutions :

   ✔ Static choice of the global distribution mode of the virtual environment

      ✗ With no explicit separation between network and graphic features

      ✗ With no possibility to change the distribution mode

   ✔ HCI software architectural models

      ✗ That do not impose a strong enough separation with 3D graphics

      ✗ That do not locate precisely the place to put the collaboration management

➢ These two research domains must be merged more efficiently

➢ We propose a new interpretation of the PAC model: PAC-C3D

  ✔ PAC + explicit interfaces for better independence between components

➢ We propose a new interpretation of the PAC model: PAC-C3D

　✔ PAC + explicit management of distribution policies
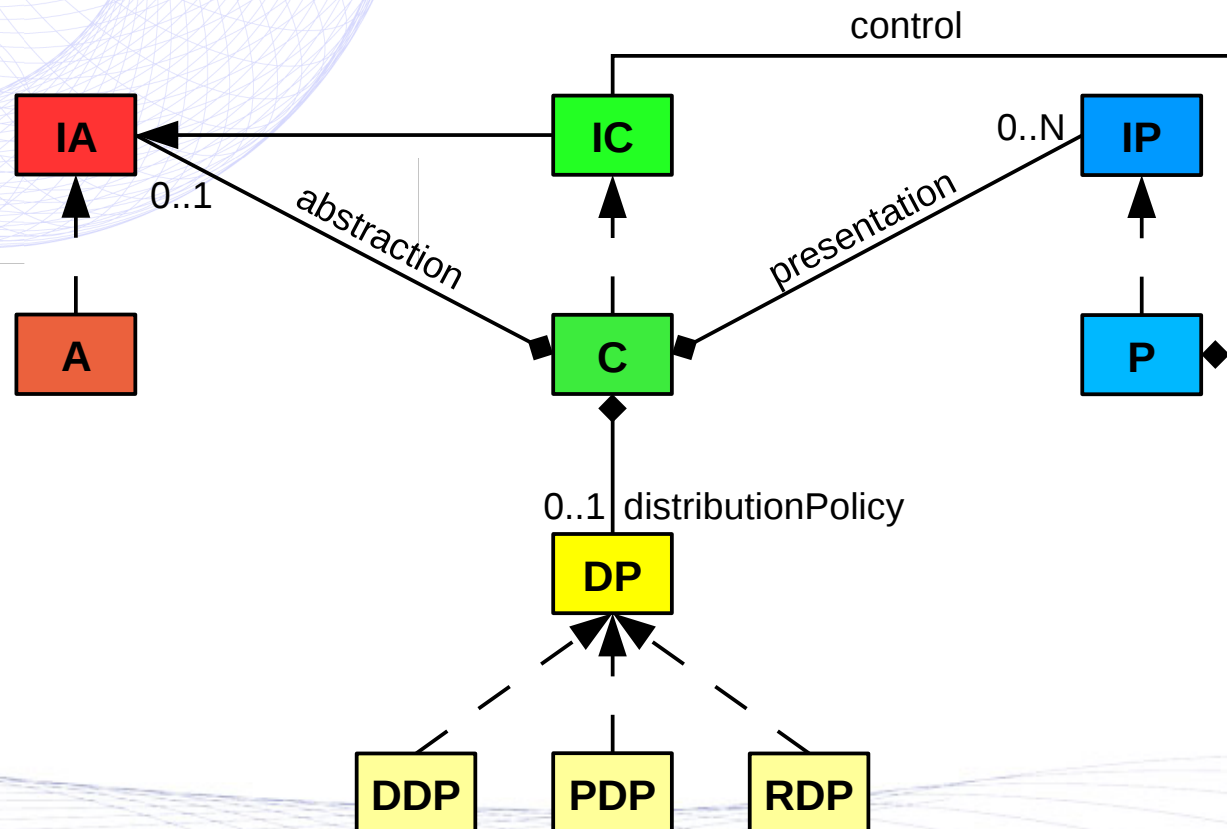
➢ PAC-C3D and duplicated architecture
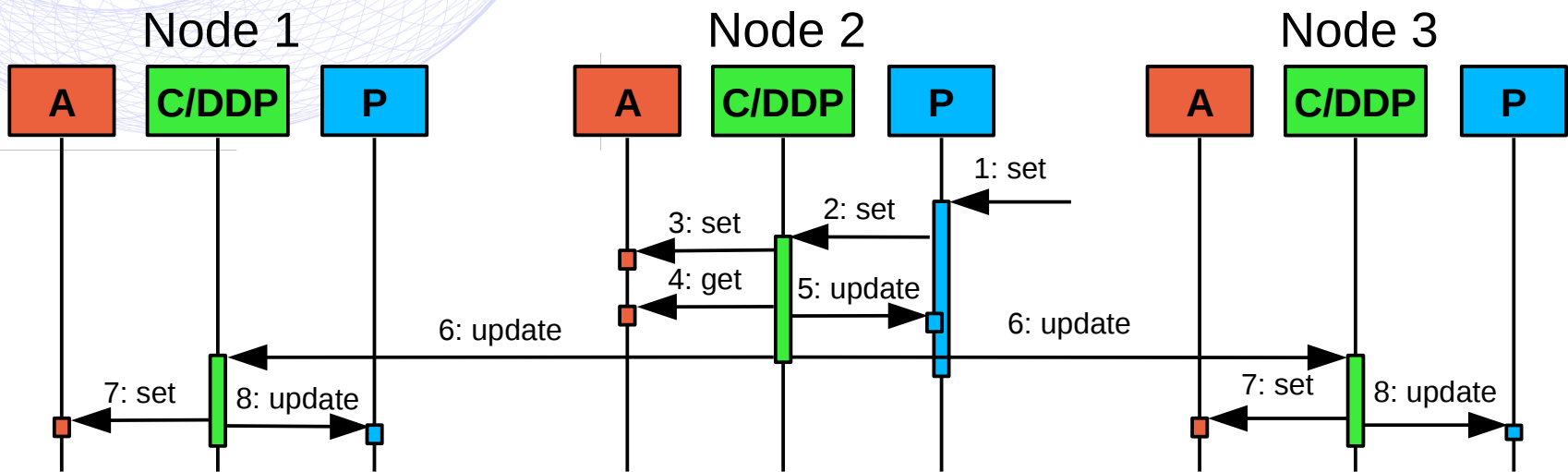
➢ PAC-C3D and centralized architecture

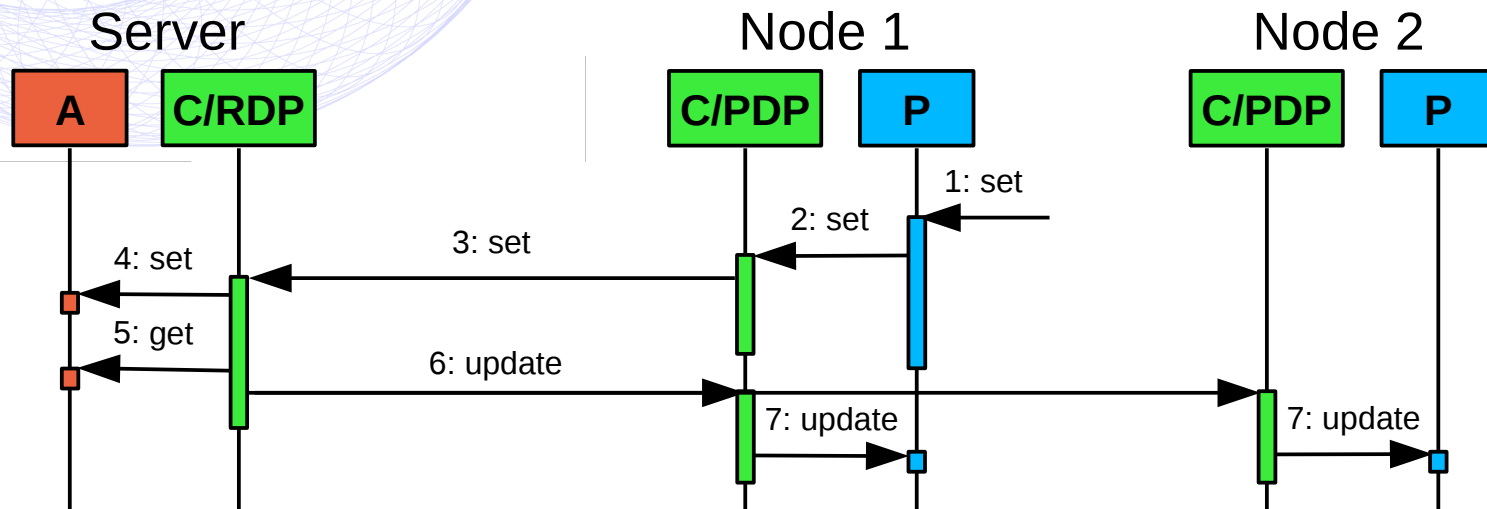➤ PAC-C3D and hybrid architecture
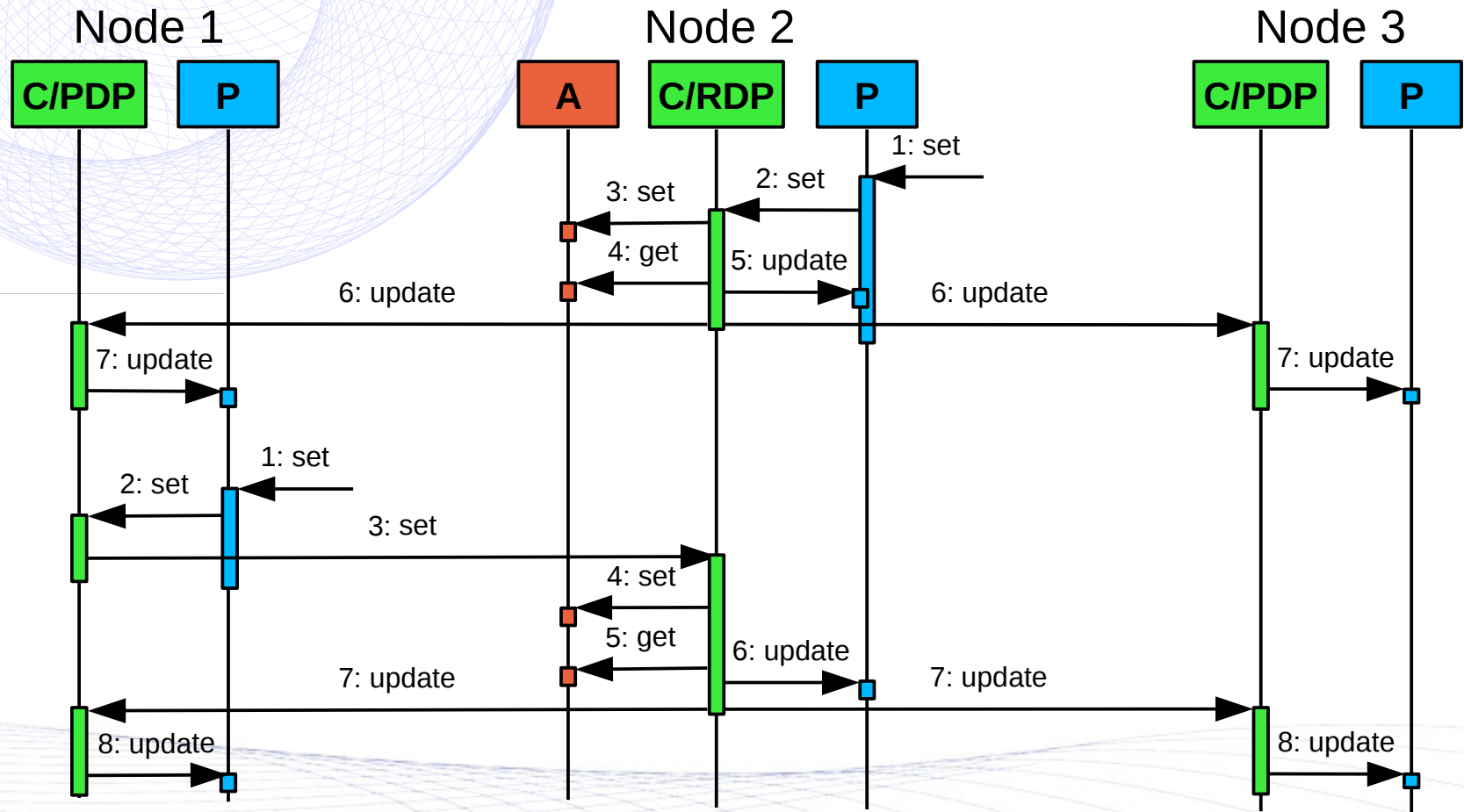
➢ PAC-C3D and AbstractFactory for creation of PAC components
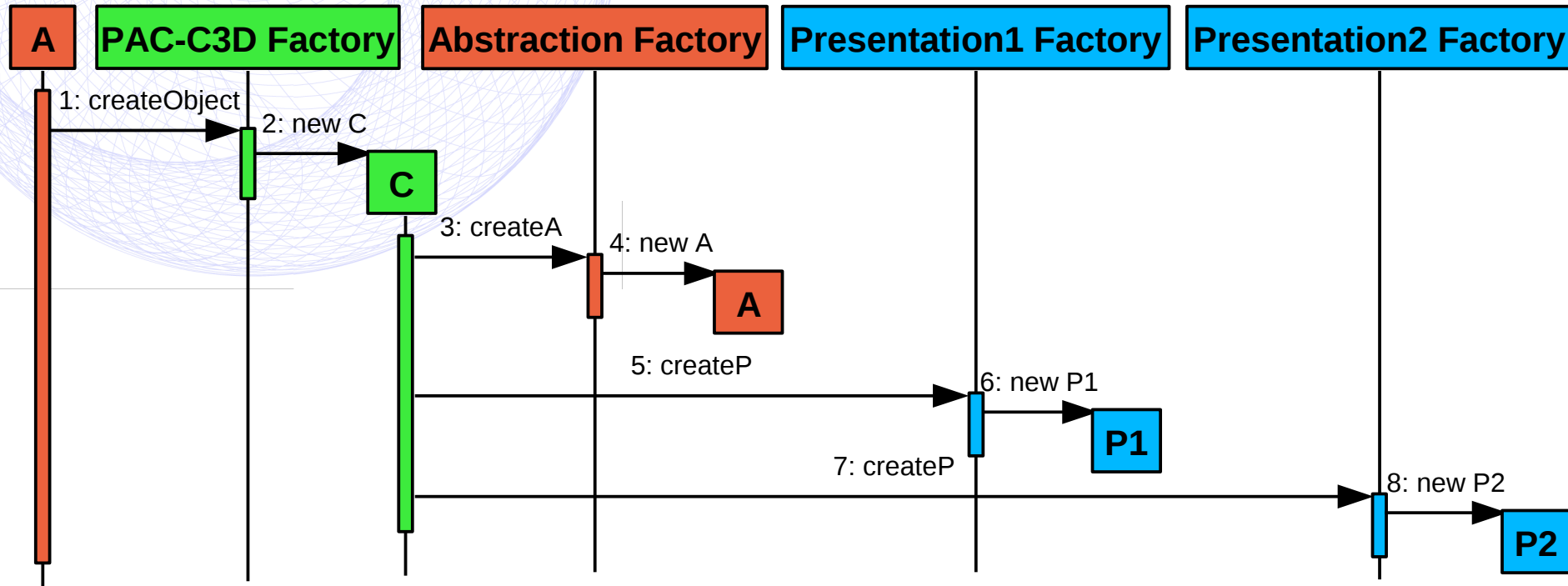
➢ Then the controller may send messages to other nodes

✔ According to its distribution policy…

➢ PAC-C3D current implementation in the Collaviz framework:

- ✔ Abstractions:
    - ✗ Java
- ✔ Controls:
    - ✗ Java + tcp or http/https communication layer + choice between 3 distribution policies
- ✔ Presentations:
    - ✗ Java3D
    - ✗ jReality
    - ✗ jMonkey (not yet fully operational)
    - ✗ Scirenderer (the Scilab 3D rendering engine)
    - ✗ jBullet (for Physics)

➢ PAC-C3D encourages devices abstraction
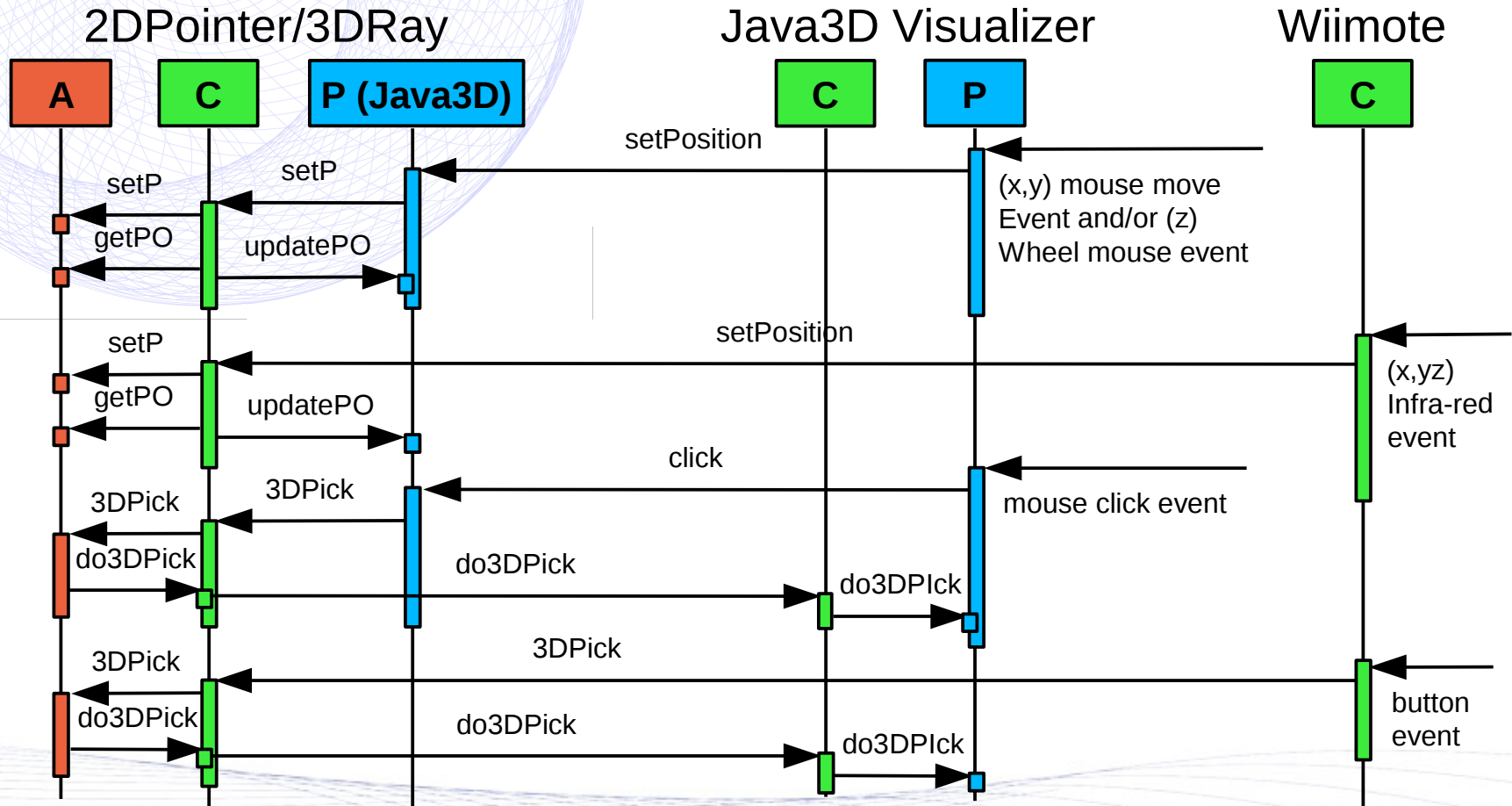
➢ Allows to drive virtual interaction tools with any physical device…

➢ PAC-C3D and devices abstraction

2DPointer/3DRay   jReality Visualizer   Wiimote

| A | C | P (jReality) | | C | P | | C |

setPosition

setP

(x,yz)
Infra-red
event

getPO

updatePO

3DPick

3DPick

button
event

do3DPick
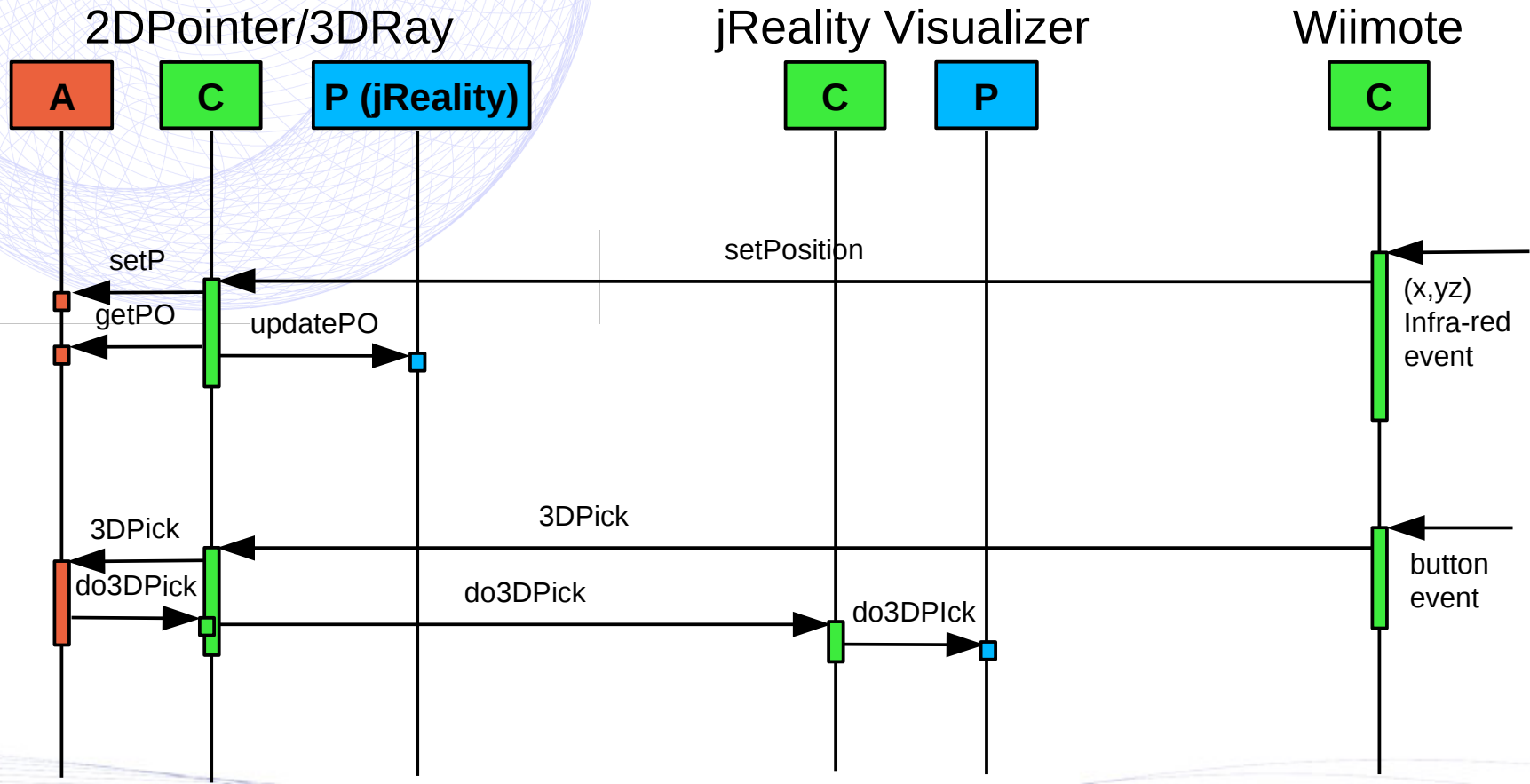
do3DPick

do3DPIck

➢ Allows not to bother about graphics inputs in jReality...

➢ PAC-C3D offers interoperability between heterogeneous 3D viewers
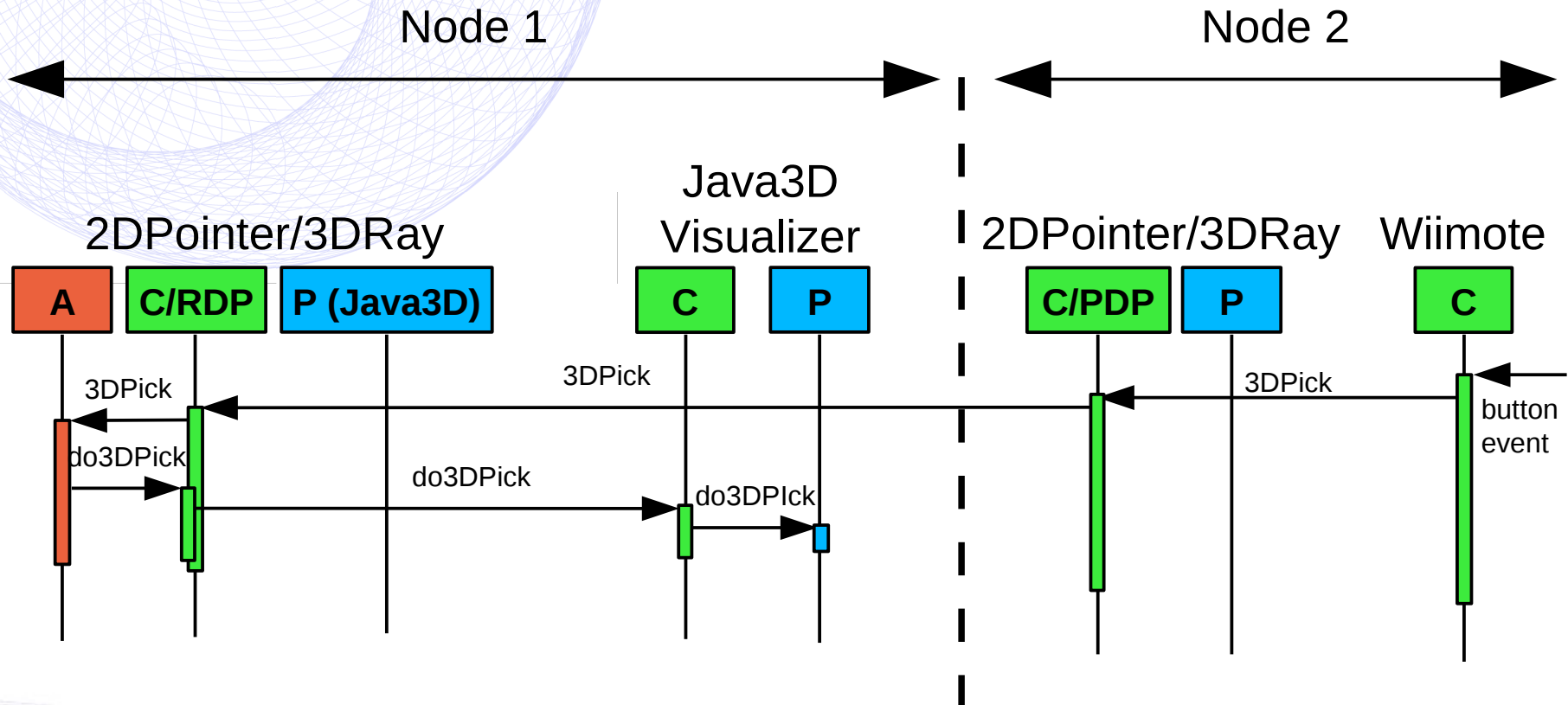
➢ Allows not to bother about any 3D picking with new 3D API...

➢ PAC-C3D and Physics engines
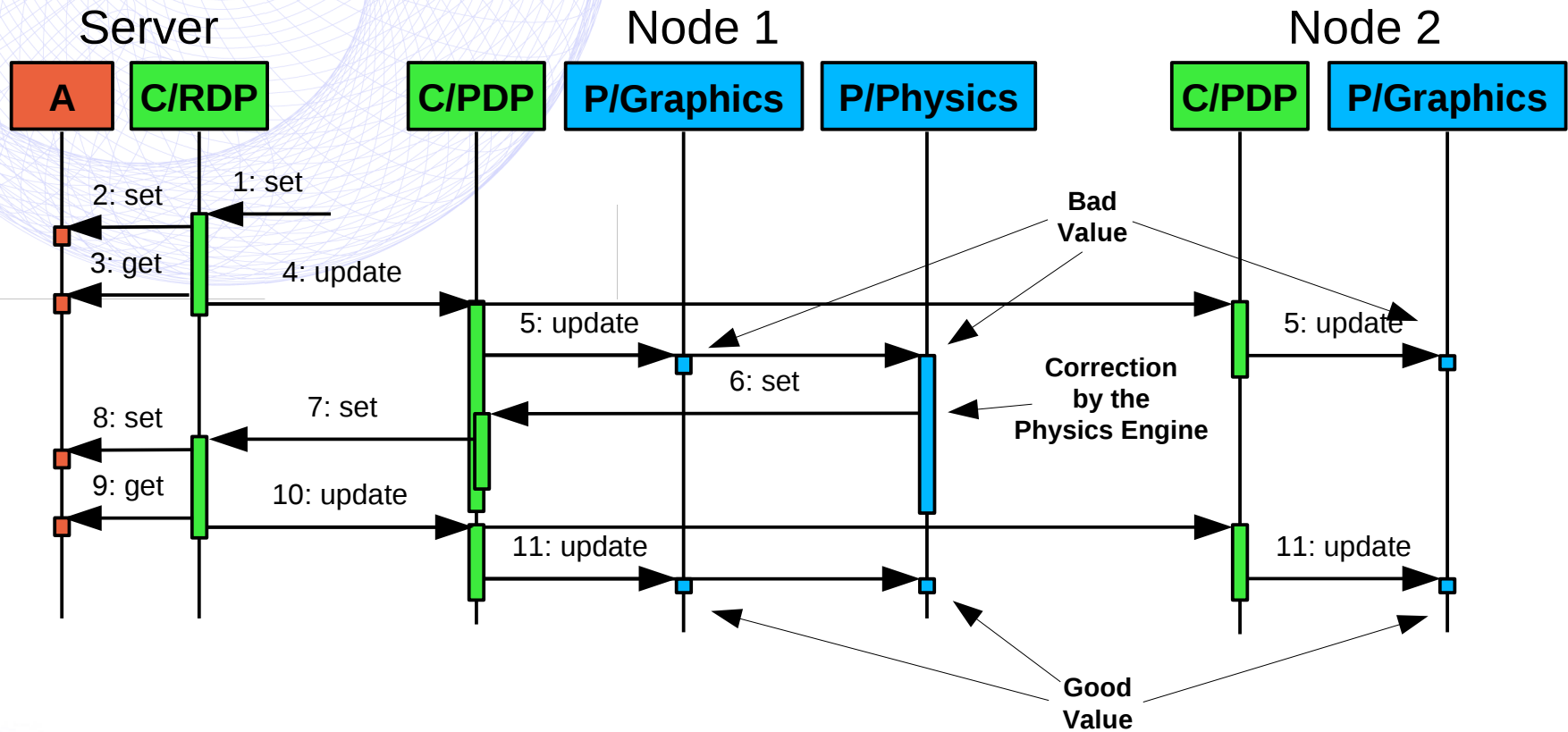
➢ Physics engines are considered as active presentation components

- PAC-C3D :
  - ✔ Evolution of the PAC model
    - ✗ Explicit interfaces between components
    - ✗ Very small dependency on 3D graphics API
    - ✗ Control components are also in charge of the collaboration
  - ✔ Deals with different distribution modes
    - ✗ Through the control components

- Tangible results through our implementation :
  - ✔ Ability to build "abstract" virtual worlds
    - ✗ With quite standardized control components
    - ✗ With dynamical choice of distribution mode, for each virtual object
    - ✗ With wide choice of 3D graphics visualization
  - ✔ Ability to provide abstract navigation and interaction tools
    - ✗ That can be driven by any kind of physical devices, even by a 2D GUI
  - ✔ Ability to share a virtual world between heterogeneous 3D viewers
    - ✗ With possibility to exchange services between them !

Future work

- Take into account "active presentations" more efficiently
  - Work in progress…

- Couple PAC-C3D objects with other kinds of "engines"
  - Artificial intelligence behavior libraries

- Propose a kind of abstract scene-graph to describe CVE
  - Should cover the most common 3D features such as X3D
  - Should describe the distribution mode of each object
  - Should deal with interactive and collaborative capabilities for virtual objects
    - Kind of interaction for each object, how many users at the same time, …
  - Should manage access rights for interaction and collaboration
    - Who can interact, who can join or interrupt an interaction…
  - Should describe the interaction tools capabilities
  - Should describe the physical environment of the users
    - To embed it within the virtual environment, to prevent collisions, to improve collaboration, …

➢ Any questions ?

   ✔ Come and see our collaborative demo tomorrow !

jReality   Java3D + jBullet

     2 wiimotes, no picking, no physics   1 mouse, 3D picking, physics